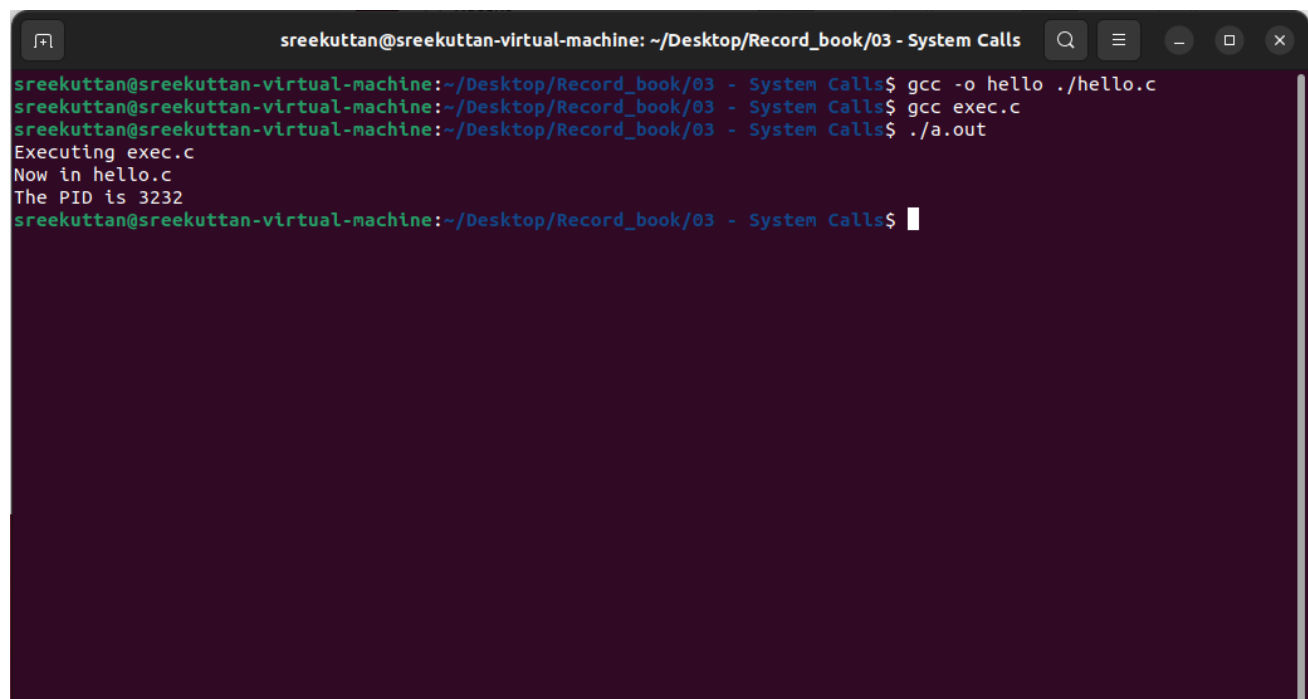


## PROGRAM

```
//exec.c
#include<stdio.h>
#include<unistd.h>
void main(){
    printf("Executing exec.c\n");
    char *args[]={"/hello",NULL};
    execv(args[0],args);
    printf("This line will not be executed");
}

//hello.c
#include<stdio.h>
#include<unistd.h>
void main(){
    printf("Now in hello.c\n");
    printf("The PID is %d\n",getpid());
}
```

## OUTPUT

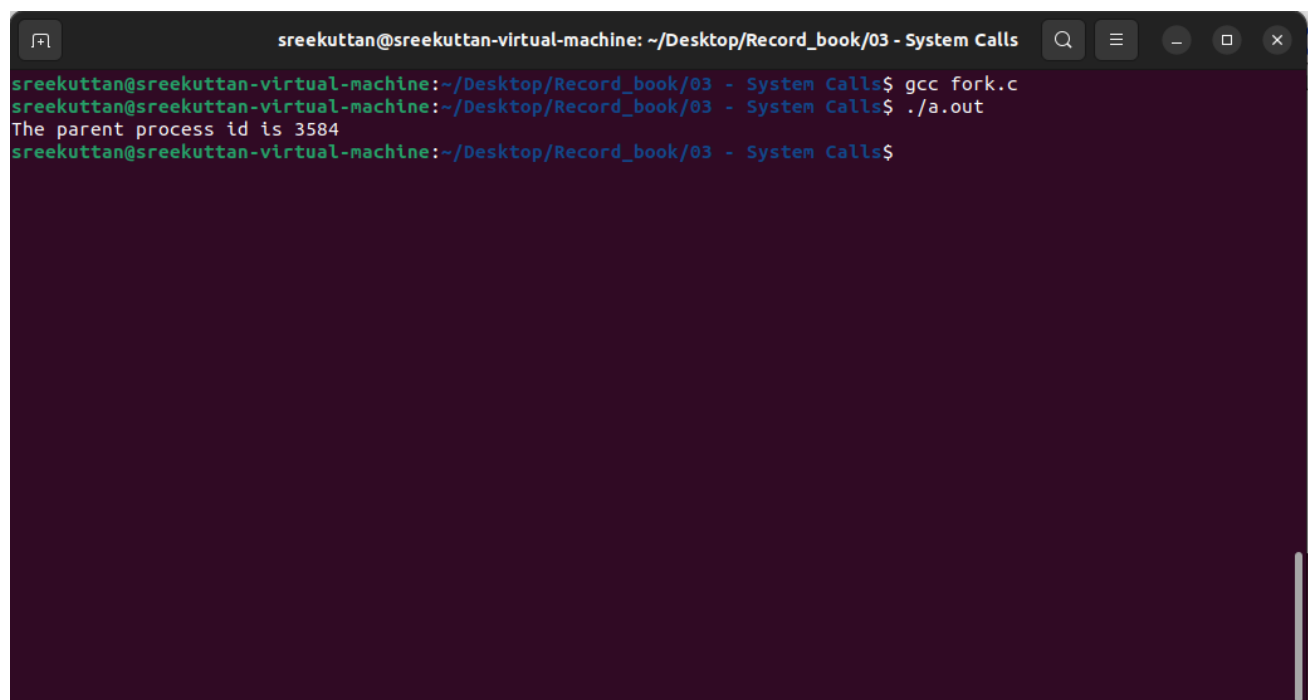
A terminal window titled 'sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record\_book/03 - System Calls'. The terminal shows the following commands and output:

```
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ gcc -o hello ./hello.c
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ gcc exec.c
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ ./a.out
Executing exec.c
Now in hello.c
The PID is 3232
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$
```

## PROGRAM

```
//fork
#include <stdio.h>
#include <unistd.h>
int main(){
    int pid,pid1,pid2;
    pid=fork();
    if(pid == 1){
        printf("Error in process \n");
    }
    if(pid !=0){
        pid1=getpid();
        printf("The parent process id is %d\n",pid1);
    }
}
```

## OUTPUT

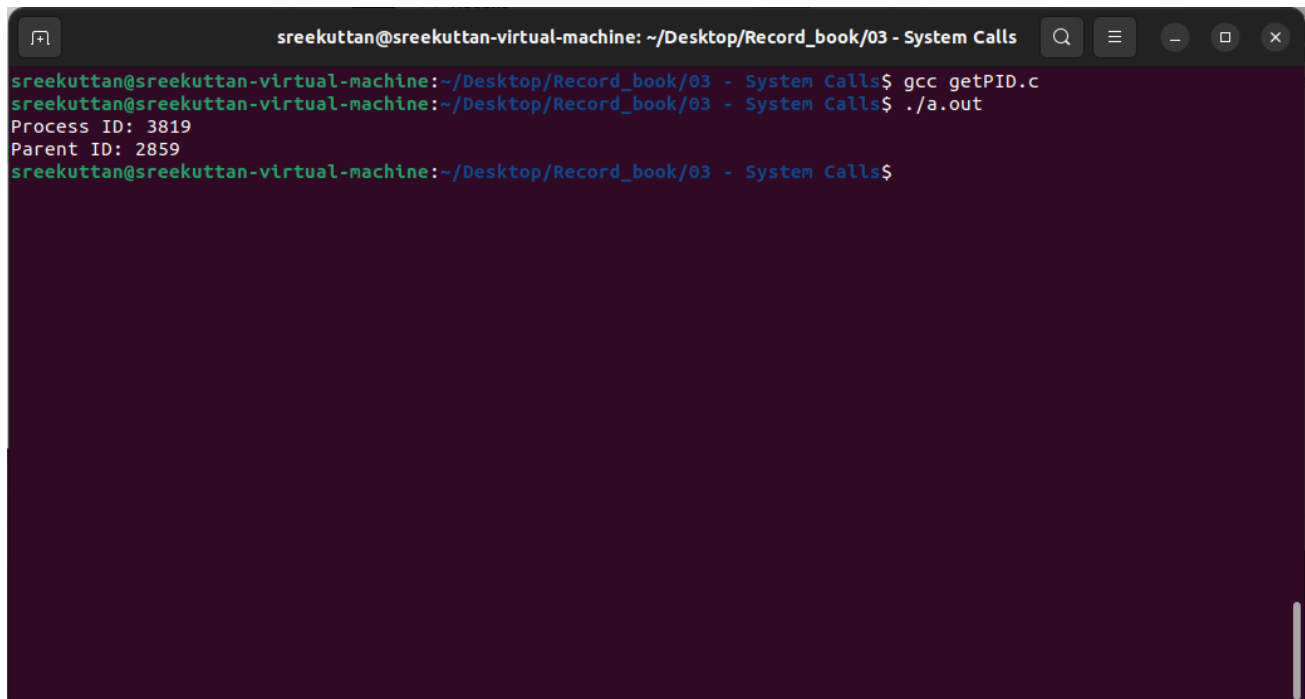


```
sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record_book/03 - System Calls
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ gcc fork.c
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ ./a.out
The parent process id is 3584
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$
```

## PROGRAM

```
//getPID
#include <stdio.h>
#include <unistd.h>
int main() {
    int pid, ppid;
    pid = getpid();
    ppid = getppid();
    printf("Process ID: %d\n", pid);
    printf("Parent ID: %d\n", ppid);
    return 0;
}
```

## OUTPUT

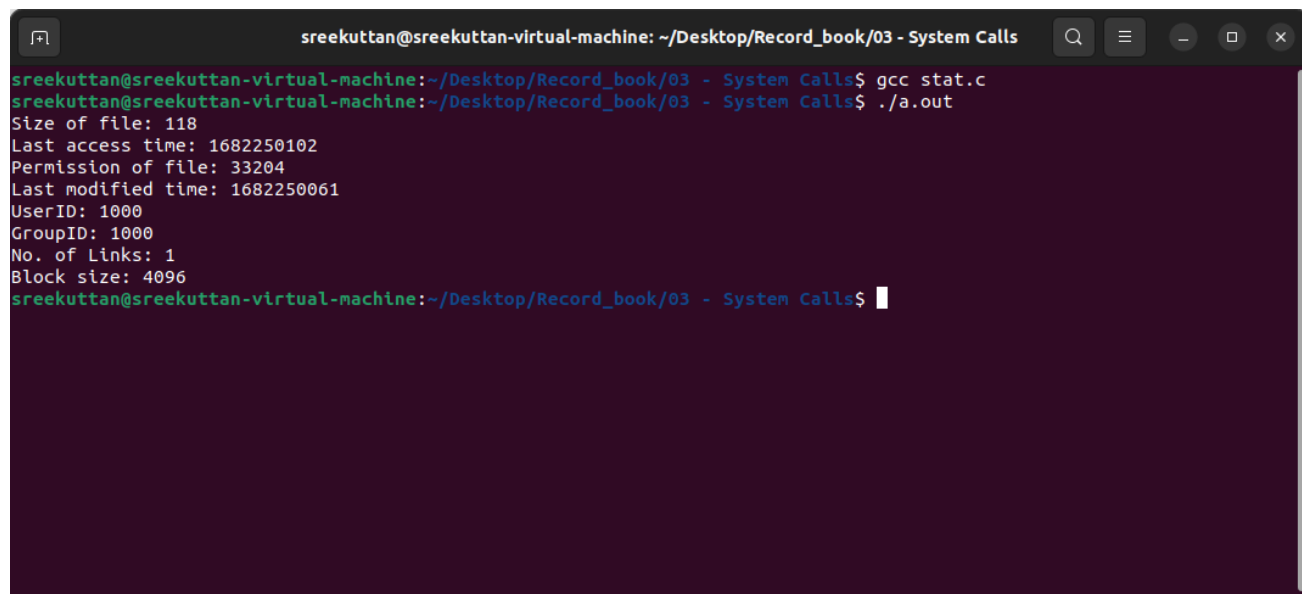
A terminal window with a dark background and light green text. The window title is "sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record\_book/03 - System Calls". The terminal shows the compilation and execution of a C program. The output displays the process ID as 3819 and the parent ID as 2859.

```
sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record_book/03 - System Calls$ gcc getPID.c
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ ./a.out
Process ID: 3819
Parent ID: 2859
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$
```

## PROGRAM

```
//stat
#include <stdio.h>
#include <sys/stat.h>
int main() {
    struct stat nfile;
    stat("hello.c",&nfile);
    printf("Size of file: %ld\n", nfile.st_size);
    printf("Last access time: %ld\n", nfile.st_atime);
    printf("Permission of file: %d\n", nfile.st_mode);
    printf("Last modified time: %ld\n", nfile.st_mtime);
    printf("UserID: %u\n", nfile.st_uid);
    printf("GroupID: %d\n", nfile.st_gid);
    printf("No. of Links: %ld\n", nfile.st_nlink);
    printf("Block size: %ld\n", nfile.st_blksize);
    return 0;
}
```

## OUTPUT

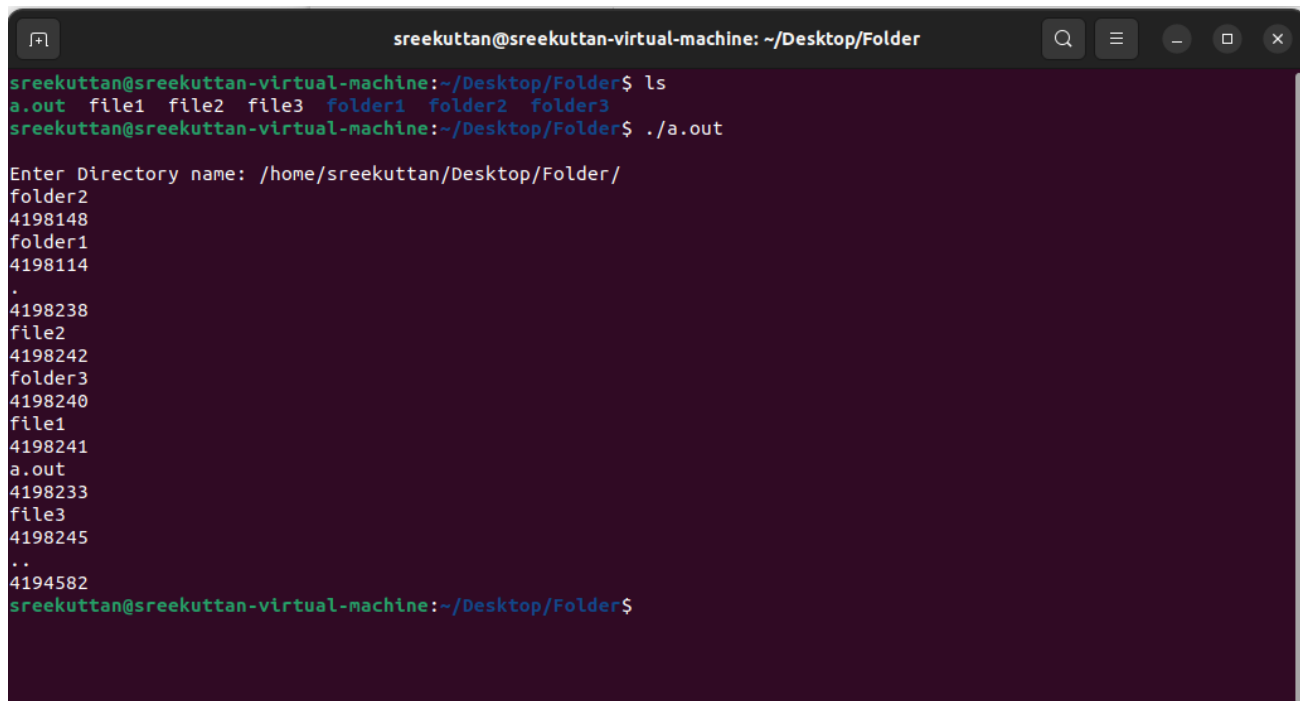
A terminal window titled "sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record\_book/03 - System Calls" shows the compilation and execution of the stat program. The user runs "gcc stat.c" and then "./a.out". The output displays file statistics for "hello.c": Size of file: 118, Last access time: 1682250102, Permission of file: 33204, Last modified time: 1682250061, UserID: 1000, GroupID: 1000, No. of Links: 1, and Block size: 4096.

```
sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record_book/03 - System Calls$ gcc stat.c
sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record_book/03 - System Calls$ ./a.out
Size of file: 118
Last access time: 1682250102
Permission of file: 33204
Last modified time: 1682250061
UserID: 1000
GroupID: 1000
No. of Links: 1
Block size: 4096
sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record_book/03 - System Calls$
```

## PROGRAM

```
//opendir readdir
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
struct dirent *dptr;
int main(int argc, char *argv[]) {
    char buff[100];
    DIR *dirp;
    printf("\n\nEnter Directory name: ");
    scanf("%s", buff);
    if ((dirp = opendir(buff)) == NULL) {
        printf("The given directory doesn't exist");
        exit(1);
    }
    while (dptr = readdir(dirp)) {
        printf("%s\n", dptr->d_name);
        printf("%d\n", dptr->d_ino);
    }
    closedir(dirp);
}
```

## OUTPUT

A terminal window titled 'sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Folder' with standard window controls. The terminal shows the execution of a C program. The user runs 'ls' showing files 'a.out', 'file1', 'file2', 'file3', 'folder1', 'folder2', and 'folder3'. Then they run './a.out'. The program prompts 'Enter Directory name: /home/sreekuttan/Desktop/Folder/'. It then lists the contents of that directory, showing each filename followed by its inode number on a new line: 'folder2' (4198148), 'folder1' (4198114), '.' (4198238), 'file2' (4198242), 'folder3' (4198240), 'file1' (4198241), 'a.out' (4198233), 'file3' (4198245), '..', and '..' (4194582). The prompt returns to 'sreekuttan@sreekuttan-virtual-machine:~/Desktop/Folder\$'.

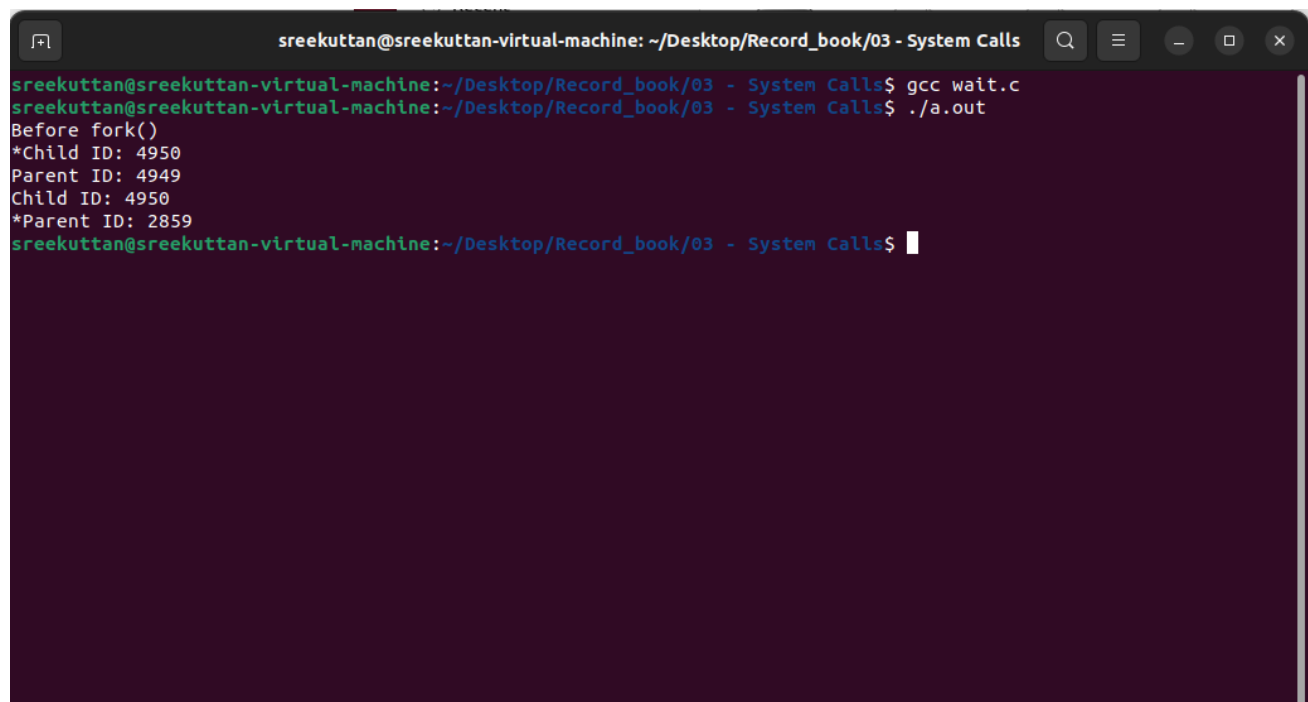
```
sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Folder
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Folder$ ls
a.out file1 file2 file3 folder1 folder2 folder3
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Folder$ ./a.out

Enter Directory name: /home/sreekuttan/Desktop/Folder/
folder2
4198148
folder1
4198114
.
4198238
file2
4198242
folder3
4198240
file1
4198241
a.out
4198233
file3
4198245
..
4194582
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Folder$
```

## PROGRAM

```
//wait
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
    pid_t p;
    printf("Before fork()\n");
    p = fork();
    if (p == 0) {
        printf("*Child ID: %d\n", getpid());
        printf("Parent ID: %d\n", getppid());
    }
    else {
        wait(NULL);
        printf("Child ID: %d\n", p);
        printf("*Parent ID: %d\n", getppid());
    }
    return 0;
}
```

## OUTPUT

A terminal window titled 'sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record\_book/03 - System Calls' with search, menu, and window control icons. The terminal shows the execution of a C program. The user runs 'gcc wait.c' and then './a.out'. The output is: 'Before fork()', '\*Child ID: 4950', 'Parent ID: 4949', 'Child ID: 4950', and '\*Parent ID: 2859'. The prompt returns to 'sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record\_book/03 - System Calls\$'.

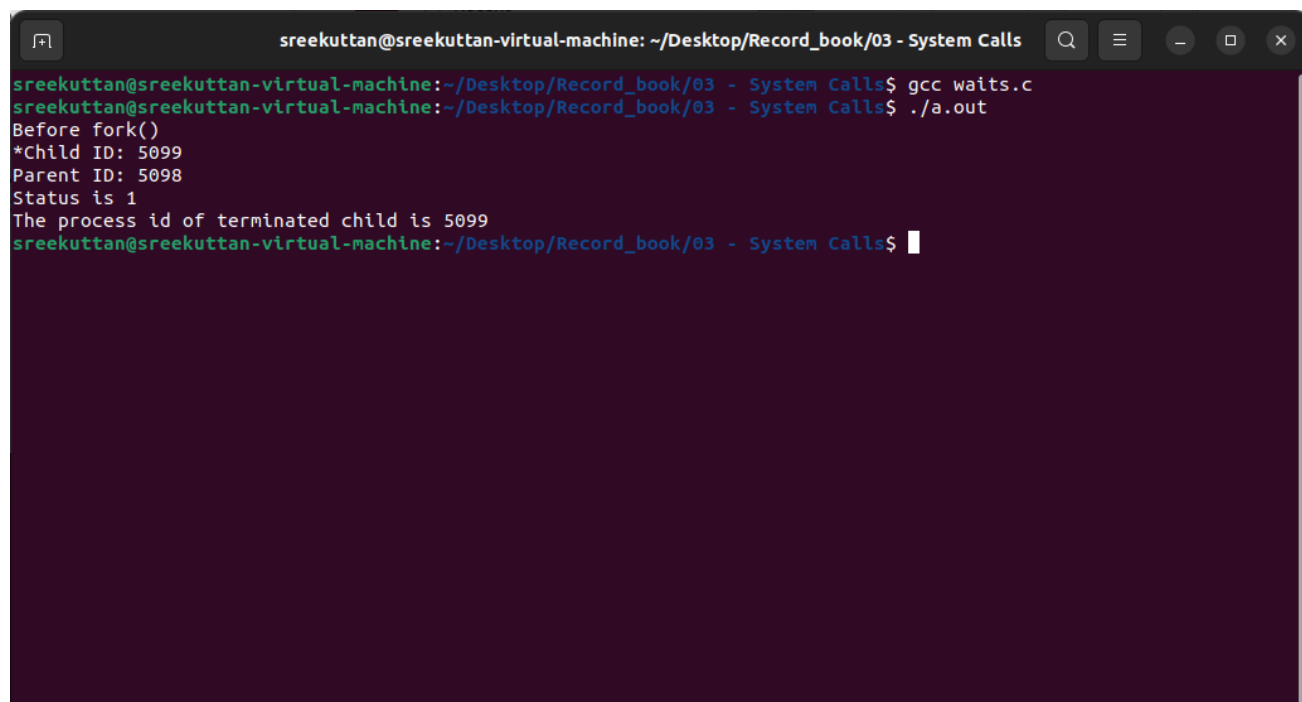
```
sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record_book/03 - System Calls$ gcc wait.c
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ ./a.out
Before fork()
*Child ID: 4950
Parent ID: 4949
Child ID: 4950
*Parent ID: 2859
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$
```

## PROGRAM

```
//waits
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
    pid_t p;

    printf("Before fork()\n");
    p = fork();
    if (p == 0) {
        printf("*Child ID: %d\n", getpid());
        printf("Parent ID: %d\n", getppid());
    }
    else {
        //w=wait(NULL);
        int wstatus;
        int w1 = wait(&wstatus);
        printf("Status is %d \n",WIFEXITED(wstatus));
        printf("The process id of terminated child is %d\n",w1);
    }
    return 0;
}
```

## OUTPUT

A terminal window titled "sreekuttan@sreekuttan-virtual-machine: ~/Desktop/Record\_book/03 - System Calls" shows the execution of the program. The user runs "gcc waits.c" and then "./a.out". The output is as follows:

```
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ gcc waits.c
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$ ./a.out
Before fork()
*Child ID: 5099
Parent ID: 5098
Status is 1
The process id of terminated child is 5099
sreekuttan@sreekuttan-virtual-machine:~/Desktop/Record_book/03 - System Calls$
```